Perancangan Permainan *Flood Filling* pada Platform Android

Hasnul Arief Fikri, Romi Fadillah Rahmat, Opim Salim Sitompul

Program Studi S1 Teknologi Informasi
Fakultas Ilmu Komputer dan Teknologi Informasi
Universitas Sumatera Utara
E-mail: fikri.hasnul@students.usu.ac.id, romi.fadillah@usu.ac.id, opim@usu.ac.id

***Abstrak*—*Flood filling* adalah *game* kombinasi warna yang dibuat menggunakan algoritma *flood fill*. Algoritma *flood fill* adalah algoritma yang menentukan area yang terhubung terhadap *node* pada *multidimensional array*. Perancangan permainan *mobile flood filling* pada platform Android ini adalah sebuah analisis, desain, dan implementasi algoritma *flood filling* dalam pembuatan *mobile game*. Pembuatan aplikasi ini bertujuan untuk menjelaskan cara kerja aplikasi dan penerapannya di lingkungan Android. Dua buah mode permainan baru juga dikembangkan agar permainan ini lebih menarik. Aplikasi ini dikembangkan dengan metode perancangan UML dan bahasa pemrograman Java. Hasil yang diperoleh adalah aplikasi permainan *flood filling* yang kompatibel untuk perangkat Android dan lulus *user requirement review* dalam perancangan *mobile game*.**

***Kata kunci*—Algoritma *flood fill*, Android, *mobile game*, permainan *flood filling*.**

I. PENDAHULUAN

Perkembangan yang pesat pada *video game* mendorong para pengembang *game* untuk mengembangkan *video game* yang lebih baik dari sebelumnya. Jenis *game* berkembang dari jenis *arcade game*, *console game*, dan *mobile game*. Diawal tahun 2010 *mobile game* berkembang sangat pesat (terutama di *smartphone*) hingga dapat mengimbangi *portable game device* sehingga *mobile game* menjadi sangat diminati [1].

Flood filling adalah sebuah permainan papan (*board game*) yang cukup menarik dalam mengasah otak pengguna. Permainan yang cukup populer di kalangan pengguna perangkat *mobile* ini dibuat menggunakan algoritma *flood fill* dan dimainkan dengan teknik kombinatorik. Pemain akan diberikan sebuah papan *n* x *n* berisi beragam warna dan diharuskan membanjiri seluruh papan hanya dengan satu warna. Dalam dunia nyata simulasi *flood filling* dapat dijumpai pada penyakit menular. Contohnya adalah bagaimana *zombie* menginfeksi non-*zombie* untuk menjadi *zombie* [2].

Flood filling merupakan permainan papan yang sangat sederhana dan dapat diselesaikan biasanya dalam waktu singkat untuk papan kecil (papan 12 x 12 dengan 6 warna). Tingkat kesulitan dalam permainan ini tergantung pada jumlah warna, ukuran papan, dan jumlah langkah maksimal yang diperbolehkan [3]. *Flood filling* juga dikenal dengan nama floodit karena pertama sekali dipasarkan dengan nama Flood-It! oleh sebuah perusahaan *web gadget* di tahun 2006.

Flood fill adalah sebuah algoritma seleksi area yang diterapkan dalam pengembangan permainan ini. Algoritma *flood fill* digunakan sebagai fungsi utama pewarnaan di setiap langkah pilihan pemain.

Algoritma *flood fill* digunakan untuk menentukan area yang terhubung terhadap *node* yang diberikan pada *multidimensional array*. Algoritma ini telah digunakan pada *tool bucket fill* di program Microsoft Paint untuk menentukan bagian mana saja dari sebuah *bitmap* untuk diisikan warna [4]. Algoritma ini membutuhkan 3 parameter yaitu *start node*, *target color*, dan *replacement color*. Algoritma mencari semua *node* dalam *array* yang terhubung ke *start node* oleh *target color* kemudian menggantinya menjadi *replacement color*.

Android adalah sistem operasi *mobile* yang cukup berkembang saat ini. Perkembangan tersebut mengakibatkan meningkatnya aplikasi-aplikasi *mobile* berbasis Android. Dengan memanfaatkan perkembangan Android diharapkan aplikasi yang dibangun dapat lebih bermanfaat dan bersifat ekonomis.

Mencermati hal-hal diatas maka kami berkeinginan untuk membuat aplikasi *flood filling* dengan menerapkan algoritma *flood fill* sebagai fungsi dasar pewarnaan. Perancangan dua buah mode permainan baru dilakukan agar permainan ini menjadi lebih menarik. Diharapkan penelitian ini dapat digunakan untuk memberi kejelasan tentang cara kerja algoritma *flood fill* dan media pembelajaran dalam perancangan permainan *mobile* khususnya Android.

Adapun tujuan penelitian ini adalah untuk menganalisis, mendesain, dan mengimplementasikan algoritma *flood fill* menjadi sebuah aplikasi *flood filling* untuk platform Android.

II. IDENTIFIKASI MASALAH

Permainan *flood filling* adalah salah satu permasalahan kombinatorik dimana pemain harus memiliki solusi, terdiri dari kombinasi warna (langkah) agar dapat menyelesaikan

papan secara efisien [3]. Setiap permainan memberikan batasan langkah maksimal yang berbeda-beda agar dapat memberikan tantangan. Jika pemain gagal menyelesaikan permainan kurang dari batasan langkah maka permainan berakhir.

Telah dibuktikan dalam mencari solusi untuk permainan Flood-It adalah *NP-hard* untuk papan dengan warna ≥ 3 dan ini juga berlaku untuk varian permainan Flood-It bernama Free-Flood-It dimana pemain dapat menentukan sendiri titik awal permainan [4].

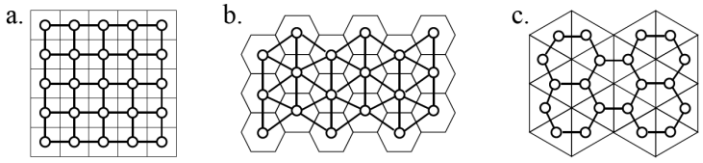
Meeks dan Scott [5] menjelaskan bahwa berdasarkan masalah permainan ini terbagi 3 yaitu :

a. FIXED-FLOOD-IT, masalah yang ditentukan berdasarkan jumlah minimum pergerakan yang diperlukan untuk membanjiri seluruh grafik warna (papan). Jika kita selalu bermain pada satu *vertex*, maka jumlah langkah bisa tak terbatas.
b. FREE-FLOOD-IT, masalahnya sama tetapi pemain dibebaskan memilih titik awal permainan.
c. *c*-FIXED-FLOOD-IT dan *c*-FREE-FLOOD-IT, merupakan varian dari FIXED-FLOOD-IT dan FREE-FLOOD-IT dimana hanya warna dari beberapa set tetap *c* yang digunakan.

Kemudian ditambahkan lagi oleh Meeks dan Scott [6] masalah terkait yang muncul ketika mempertimbangkan algoritma untuk floodit agar mempertimbangkan jumlah langkah yang diperlukan untuk menyambungkan rangkaian simpul adalah :

k-LINKING-FLOOD-IT, merupakan masalah dimana diberikan bagian *U* dari simpul *k*, untuk menentukan jumlah langkah yang diperlukan untuk membuat komponen *monochrome* yang mengandung *U*. Jumlah warna bisa tak terbatas.

Kemudian berdasarkan sel permainannya permainan ini terbagi 3 yaitu sel kotak, sel segi enam dan sel segi tiga [7]. Gambar 1 menunjukkan ilustrasi ketiga jenis papan dengan selnya masing-masing.



Gambar 1. (a) sel kotak, (b) sel segi enam, dan (c) sel segi tiga

Permasalahan dalam perancangan permainan *flood filling* pada platform Android ini adalah bagaimana merancang dan menerapkan algoritma *flood fill* dalam pengembangan *flood filling game* dan implementasinya di lingkungan Android. Permainan yang kami bangun berjenis fixed-flood-it yaitu jenis permainan dengan permasalahan mencari pergerakan minimum dengan titik awal pembanjiran statis. Kemudian sel yang digunakan adalah sel kotak karena papan yang dibentuk dibuat dengan matriks dan arah pewarnaan algoritma *flood fill* yang digunakan adalah 4 arah, barat, timur, utara, dan selatan.

III. PENELITIAN TERDAHULU

Penelitian terkait mengenai permainan *flood filling* adalah berkaitan dengan tingkat kompleksitas dari penyelesaian permainan itu sendiri.

Arthur *et al* [4] menemukan bahwa dalam mencari solusi permainan *flood filling* adalah NP-hard untuk set warna c ≥ 3, hal ini juga berlaku saat pemain dapat menentukan sendiri titik awal pewarnaan (Free-Flood-It).

Meeks dan Scott [5] mendemonstrasikan *polynomial time algorithm* untuk menghitung langkah minimum yang diperlukan untuk menyambungkan setiap pasang verteks.

Meeks dan Scott [6] melanjutkan penelitian sebelumnya, mendapat bahwa jumlah langkah minimum yang diperlukan untuk membanjiri graf G sama dengan minimum oleh jumlah langkah *spanning tree* T yang dibuat dari graf G. Hasil ini kemudian diimplementasikan untuk mendapat dua buah *polynomial time algorithm* dalam permasalahan flood filling.

Lagoutte *et al* [7] mensintesis dan memperluas hasil tentang bagaimana menyelesaikan permainan Flood-it pada siklus dengan menghitung ketinggian poset dan menyelesaikan Free-Flood-it dengan menghitung radius graf.

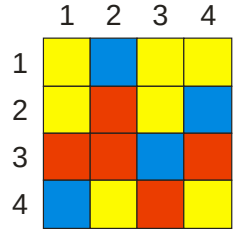
IV. METODE PENELITIAN

A. Dekripsi Algoritma Flood Fill

Secara umum urutan langkah yang dilakukan algoritma untuk melakukan pewarnaan adalah sebagai berikut :

```
begin
Deklarasi node sebagai titik awal permainan.
Deklarasi target-color sebagai warna pada node,
target-color=color(node)
Deklarasi replacement-color sebagai warna masukan pemain.
Deklarasi antrean kosong Q dan isi dengan node, Q={node}
while(!Q.isempty()) do
    buang elemen pertama Q dan set sbagai n:
    n = Q.dequeue()
    if(color(n)=target-color) then
        ubah warna pada n menjadi replacement-color:
        color(n)=replacement-color
        tambahkan titik barat, timur, utara, dan selatan dari n ke Q:
        Q.enqueue(west of n)
        Q.enqueue(east of n)
        Q.enqueue(north of n)
        Q.enqueue(south of n)
    end if
end while
end
```

Berikut adalah contoh pewarnaan menggunakan *flood fill*. Misalkan sebuah papan permainan dengan ukuran 4 x 4, 3 warna dan node (titik start) pada titik 1,1 (kiri-atas) sebagai berikut :



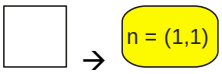
Gambar 2. Keadaan awal papan permainan

Jika warna yang dipilih pertama adalah warna biru (*replacement-color*) maka langkah-langkahnya adalah :

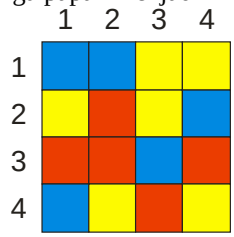
1. Sebuah antrean kosong dibuat untuk menginisialisasi ruang penelusuran pada papan permainan *flood filling* dan *target-color* diisi dengan warna pada titik start yaitu kuning

2. Masukkan node ke dalam antrean

3. Cek apakah antrean kosong atau tidak. Antrean sekarang berisi satu elemen yaitu kordinat *node*.
4. Keluarkan elemen pertama antrean dan set sebagai *n*. Antrean sekarang kosong dan sebuah variable penampung telah dibuat.



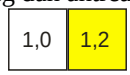
5. Cek apakah warna pada *n* sama dengan *target-color*. Jika ya sama-sama kuning maka set warna n (1,1) menjadi warna biru sehinnga papan menjadi



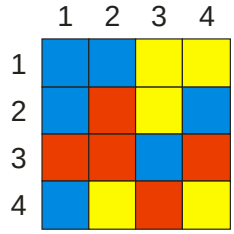
Gambar 3. Pewarnaan papan tahap 1

tambahkan 4 elemen (x+1,y), (x-1,y), (x,y+1), (x,y-1) dari *n* ke dalam antrean. Elemen yang akan dimasukkan adalah (2,1), (0,1), (1,2), dan (1,0) sehingga antrean menjadi.

6. Kemudian ulangi langkah 4 yaitu keluarkan elemen pertama antrean yaitu kordinat (2,1) dan cek apakah sama dengan *target-color*. Jika tidak maka hanya akan dibuang saja. Begitu juga dengan elemen berikutnya yaitu kordinat(0,1) sehingga pada langkah ini 2 elemen langsung dapat dibuang dan antrean menjadi



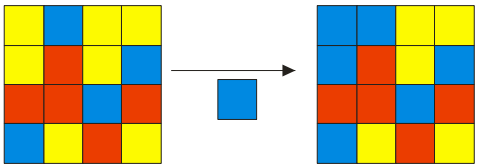
7. Ulangi lagi langkah 4 dengan mengeluarkan elemen pertama antrean yaitu kordinat (1,2) dan cek. Jika ya, maka ubah papan dan tambahkan 4 elemen sehingga papan menjadi



Gambar 4. Pewarnaan papan tahap 2

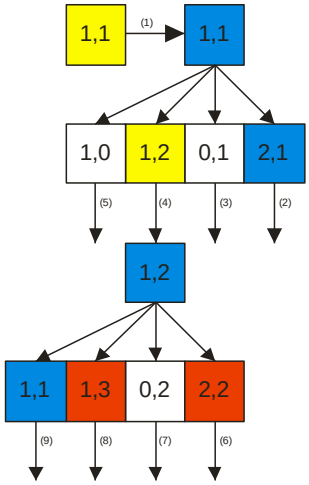
dan antrean menjadi

8. Langkah berikutnya sama dengan mengulangi langkah ke-4. Dapat dilihat dari antrean tidak ada warna yang sama dengan *target-color* yaitu warna kuning sehingga setiap elemen hanya akan dikeluarkan tanpa dilakukan pewarnaan dan penambahan elemen baru ke dalam antrean. Sehingga antrean akan kosong dan proses kerja algoritma akan berhenti.



Gambar 5. Keadaan awal dan keadaan akhir papan setelah satu langkah pewarnaan

Ilustrasi singkat dalam pohon penelusuran dapat dilihat pada gambar 6.



Gambar 6. Pohon penelusuran langkah algoritma

Dapat diambil kesimpulan bahwa diperlukan 9 kali iterasi algoritma dalam satu langkah pewarnaan untuk mengubah 2 sel warna pada papan contoh. Penambahan panjang antrean adalah 4 kali dari jumlah elemen yang dibuang yang warnanya sama dengan *target-color*.

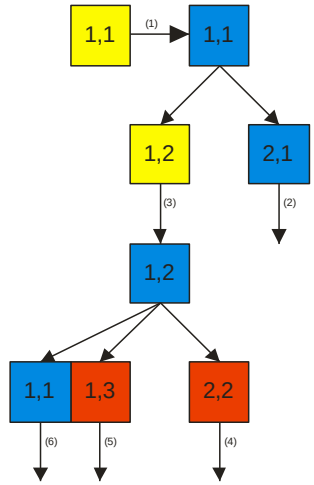
Aplikasi permainan yang dibangun akan dibuat pada *mobile platform*. Seperti diketahui dalam pengembangan aplikasi *mobile*, keterbatasan memori adalah hal yang harus diatasi. Langkah yang dapat diambil adalah dengan mengurangi panjang antrean semaksimal mungkin sehingga mengurangi kerja algoritma dalam melakukan iterasi pengecekan terhadap masing-masing elemen antrean.

B. Pemotongan Tahap 1

Langkah pertama yang dapat diambil adalah dengan menghilangkan kordinat kosong pada pohon penelusuran. Dengan menghilangkan kordinat kosong, panjang elemen antrean dapat dikurangi hingga 2 kali panjang dan lebar papan, sama dengan kelililing papan.

Berikut adalah ilustrasi pohon penelusuran setelah dilakukan pemotongan terhadap kordinat kosong.

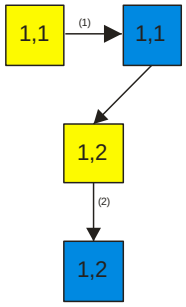
Pada pohon penelusuran kordinat kosong diwakili oleh sel berwarna putih. Dengan menghapus sel berwarna putih hanya diperlukan 6 kali iterasi algoritma untuk melakukan kerja serupa.



Gambar 7 Hasil pemotongan tahap 1

C. *Pemotongan Tahap 2*

Pemotongan tahap 2 dilakukan dengan menghilangkan percabangan yang tidak sama dengan target-color. Setiap elemen yang akan dimasukkan ke dalam antrean akan dicek terlebih dahulu apakah warnanya sama dengan target-color. Jika ya elemen akan dimasukkan, jika tidak elemen akan langsung dibuang tanpa dimasukkan ke dalam antrean sehingga panjang antrean akan semakin berkurang dan iterasi juga semakin berkurang.



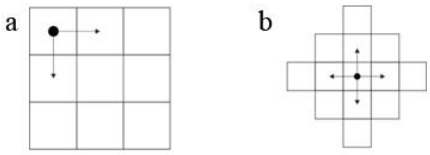
Gambar 8 Hasil pemotongan tahap 2

Dari hasil pemotongan tahap 2 didapat hanya diperlukan 2 kali iterasi algoritma untuk melakukan proses kerja serupa yaitu pewarnaan 2 sel. Dari hasil analisis ini maka *pseudocode* yang didapat adalah :

```
begin
Deklarasi node sebagai titik awal permainan.
Deklarasi target-color sebagai warna pada node,
target-color=color(node)
Deklarasi replacement-color sebagai warna masukan pemain.
Deklarasi antrean kosong Q dan isi dengan node, Q={node}
while(!Q.isempty()) do
    buang elemen pertama Q dan set sebagai n:
    n = Q.dequeue()
    ubah warna pada n menjadi replacement-color:
    color(n)=replacement-color
    tambahkan titik barat, timur, utara, dan selatan dari n ke Q:
    if(color(west of n)=target-color) then
        Q.enqueue(west of n)
    end if
    if(color(east of n)=target-color) then
        Q.enqueue(east of n)
    end if
    if(color(north of n)=target-color) then
        Q.enqueue(north of n)
    end if
    if(color(south of n)=target-color) then
        Q.enqueue(south of n)
    end if
end while
end
```

D. *Perancangan Sistem*

Permainan yang dibangun adalah *one player-casual games* dengan nama floodit. Berjalan pada resolusi 480x320 untuk perangkat Android 2.2 (Froyo) keatas. Permainan yang dirancang berjenis *Fixed-Flood-It* dengan tipe sel kotak dan alur pewarnaan 4 arah. Papan yang dibuat berbentuk kotak (titik *start* kiri atas) dan bentuk wajik (titik *start* tengah).



Gambar 9. Ilustrasi bentuk papan (a) kotak dan (b) wajik

Dua buah mode permainan yang dirancang diberi nama *credit mode* dan *time survival*. Penjelasan mengenai mode permainan dapat dilihat dari tabel berikut.

Tabel 1. Aturan mode *credit*

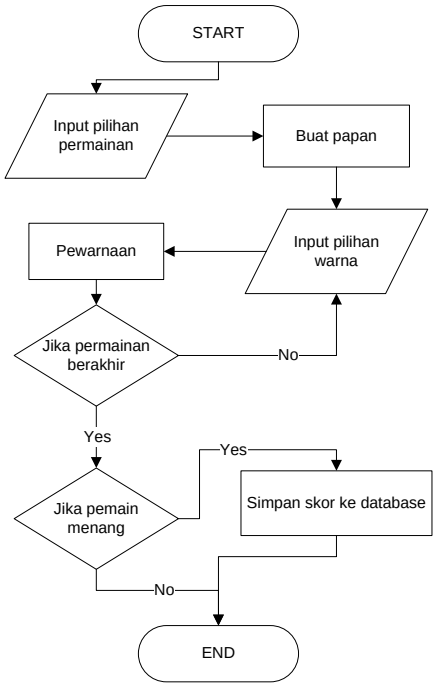
Credit game	
Pilihan papan	Kotak dan wajik
Pilihan tingkat kesulitan	Mudah, menengah, sulit
Pra kondisi	Pemain diberikan kredit langkah dan papan sesuai pilihan papan dan pilihan tingkat kesulitan.
Aliran permainan	1. Setiap pemain melangkah kredit berkurang satu. 2. Setiap pemain melangkah pemain akan mendapatkan nilai berdasarkan jumlah kotak yang dibanjiri. Nilai tersebut akan menjadi kredit agar pemain tetap dapat melanjutkan permainan.
Post kondisi	Permainan berakhir jika : 1. Menang : papan selesai dengan kredit ≥ 0 2. Kalah : kredit < 0.
Basis data skor	Penyimpanan pada basis data adalah jumlah langkah permainan dan jumlah kredit sisa. Penyimpanan pada basis data diurut berdasarkan jumlah langkah dikurang kredit secara menaik.

Tabel 2 Aturan mode *time survival*

Time survival	
Pilihan papan	Kotak dan wajik
Pilihan tingkat kesulitan	Tidak ada
Pra kondisi	1. Pemain diberikan papan dengan ukuran 5 x 5 dan jumlah warna 4. 2. Pemain diberikan waktu sebanyak 30 detik.
Aliran permainan	1. Pemain berusaha menyelesaikan papan sebanyak mungkin dengan jumlah langkah

	sesedikit mungkin. 2. Setiap selesai satu papan akan ada bonus waktu sebesar 10 detik. 3. Setiap satu papan selesai papan baru akan diberikan dengan penambahan satu warna dari sebelumnya. Jika warna sudah delapan maka berikutnya lebar papan akan bertambah dua untuk setiap papan selesai. Papan tidak bertambah panjang jika panjang sisinya = 23.
Post kondisi	Permainan berakhir jika sisa waktu = 0.
Basis data skor	Penyimpanan pada basis data adalah jumlah langkah permainan dan jumlah papan yang berhasil diselesaikan. Penyimpanan pada basis data diurutkan berdasarkan jumlah papan selesai atau jumlah langkah terbesar jika papan selesai sama secara menurun.

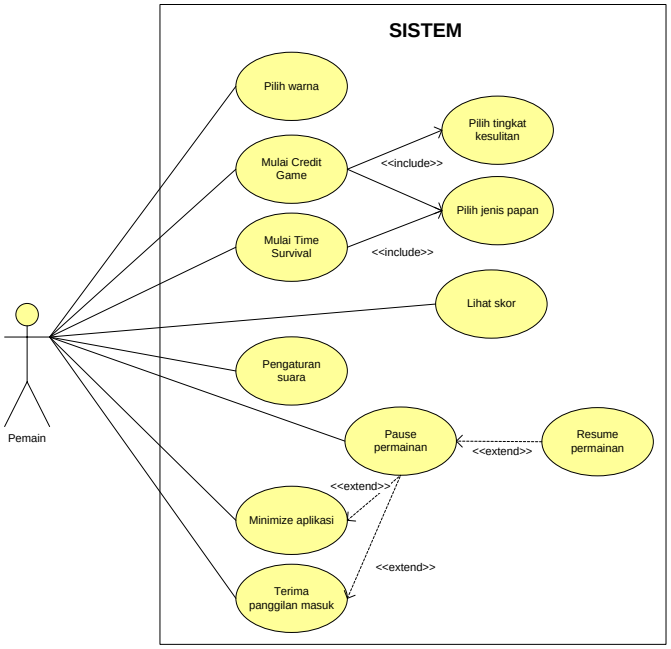
Alur permainan dimulai dengan pemain memilih mode permainan. Berdasarkan hal tersebut aplikasi akan membuat papan. Papan yang dibuat kemudian akan dimainkan sesuai mode permainan masing-masing. Jika permainan berakhir dengan catatan pemain menang maka skor akan disimpan ke database.



Gambar 10. Alur dasar permainan floodit

Pada penelitian ini digunakan UML sebagai bahasa pemodelan untuk mendesain dan merancang aplikasi permainan. Model UML yang digunakan adalah *use case diagram*, *class diagram*, dan *sequence diagram*.

Dari gambar 11 dapat dilihat bahwa hanya terdapat satu Aktor yang berhubungan dengan sistem yaitu pemain dan terdapat *Use Case* yaitu pilih warna, mulai *credit game*, mulai *time survival*, pilih tingkat kesulitan, pilih jenis papan, lihat skor, pengaturan suara, *pause* permainan, *resume* permainan, *minimize* aplikasi, dan terima panggilan masuk. Di dalam *Use Case* mulai *credit game* pemain dapat atau <<include>> memilih tingkat kesulitan dan jenis papan sedangkan di dalam *Use Case* mulai *time survival*, pemain hanya memilih jenis papan. *Use Case resume* permainan merupakan perluasan atau <<extend>> dari *Use Case pause* permainan yang dilakukan oleh pemain. *Use Case pause* permainan juga merupakan perluasan atau <<extend>> dari *Use Case minimize* aplikasi dan *Use Case* terima panggilan masuk dimana kegiatan dilakukan langsung oleh sistem.

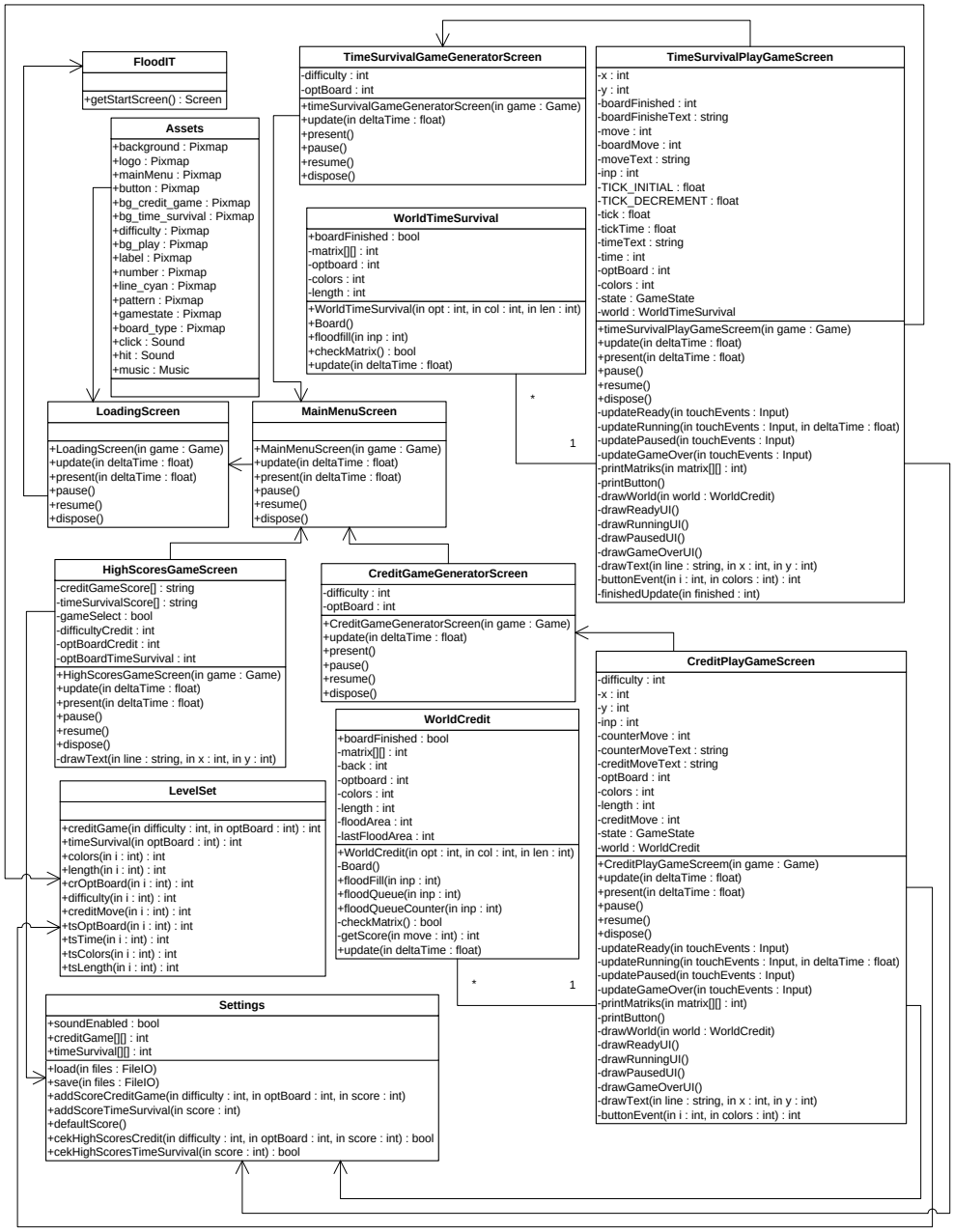


Gambar 11. *Use case diagram* floodit

Class diagram dirancang untuk menentukan objek-objek yang dibutuhkan untuk perancangan sistem. Pada perancangan *class diagram* dibutuhkan 13 kelas dengan fungsinya masing-masing dalam mengkordinir kerja sistem. Gambar *class diagram* dapat dilihat pada gambar 12.

Kelas-kelas pada gambar 12 memiliki kegunaan sebagai berikut :

1. FloodIT, merupakan kelas utama. Kelas pertama yang dipanggil saat sistem dijalankan.
2. Assets, deklarasi variable-variabel aset.
3. LoadingScreen, tempat pengisian variabel aset dan pemuatan pengaturan permainan
4. MainMenuScreen, layar menu utama.
5. CreditGameGeneratorScreen, layar pemilihan papan dan tingkat kesulitan untuk permainan credit game.
6. CreditGamePlayScreen, layar permainan mode credit game.
7. WorldCredit, generator permainan credit game.
8. TimeSurvivalGeneratorScreen, layar pemilihan papan untuk permainan time survival.
9. TimeSurvivalPlayScreen, layar permainan mode time survival.
10. WorldTimeSurvival, generator permainan time survival.
11. LevelSet, pengaturan tingkat kesulitan kedua mode permainan (khusus pengembang).
12. HighScoresGameScreen, layar papan skor.
13. Settings, penyimpanan dan pemuatan basis data skor.

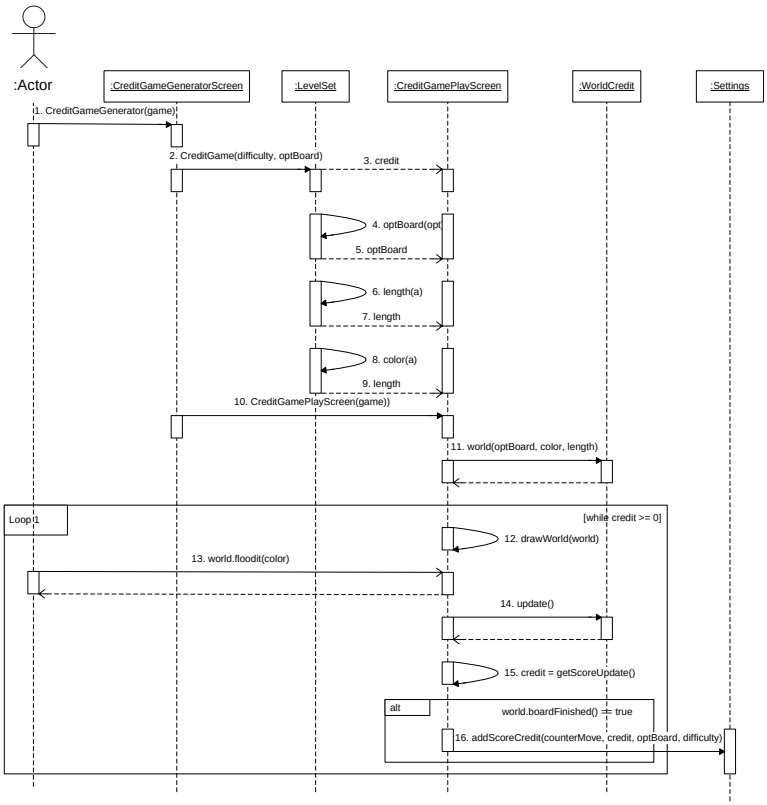


Gambar 12 *Class diagram* floodit

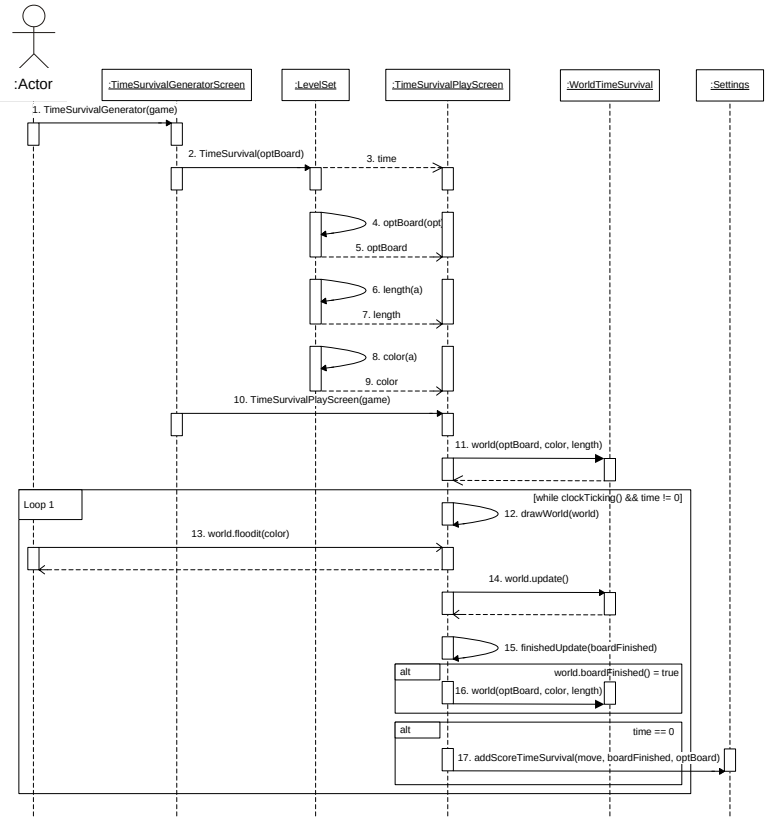
Sequence diagram digunakan untuk menggambarkan proses kerja dari masing-masing mode permainan. Berturut-turut langkah kerja mode *credit game* dan *time survival* dapat dilihat pada gambar 13 dan 14.

Pada gambar 13 jika pemain ingin memainkan permainan *credit game* maka pemain harus memilih jenis papan dan tingkat kesulitan sehingga generator permainan akan membuat papan berdasarkan pilihan. Saat papan telah selesai dibuat pemain dapat bermain dengan memilih satu dari pilihan warna yang ada. Sistem akan mengupdate papan berdasarkan input warna pemain dan mengembalikan skor yang akan dikonversi menjadi kredit langkah. Pemilihan warna oleh pemain akan terus diulang hingga kredit habis atau papan selesai.Jumlah langkah dan sisa kredit akan disimpan ke database jika pemain berhasil menyelesaikan papan.

Pada gambar 14 jika pemain ingin memainkan permainan *time survival* maka pemain harus memilih jenis papan sehingga generator permainan akan membuat papan berdasarkan pilihan dan mengatur waktu. Saat papan telah selesai dibuat pemain dapat bermain dengan memilih satu dari pilihan warna yang ada. Sistem akan mengupdate papan berdasarkan input warna pemain selama waktu belum habis. Jika papan selesai maka sistem akan menyimpan skor papan dan akan membuat papan baru untuk diselesaikan. Jika waktu telah habis sistem akan memasukkan skor papan ke database.



Gambar 13 *Sequence diagram credit game*

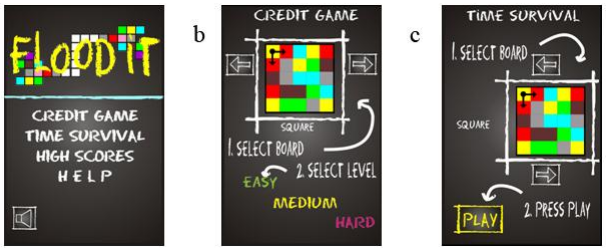


Gambar 14 *Sequence diagram time survival*

V. HASIL DAN PEMBAHASAN

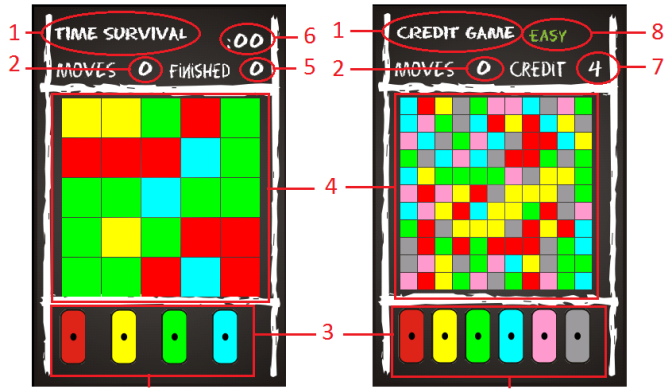
Hasil implementasi aplikasi ini dapat di-*download* pada link **http://bit.ly/hasnul-floodit-rcfinal.** File di-*upload* di website mediafire.com dalam bentuk paket instalasai Android *.apk.

Ketika pemain menjalankan aplikasi, pemain dapat memilih untuk bermain *credit game*, *time survival*, ataupun melihat skor. Gambar 15 adalah tampilan dari menu utama dan pilihan permainan *credit game* dan *time survival*.



Gambar 15. Tampilan (a) menu utama dan pilihan permainan (b) *credit game* dan (c) *time survival*

Saat pemain telah memilih permainan, sistem akan membuat papan sesuai pilihan pemain. Tampilan dari papan dapat dilihat pada gambar 16.



Gambar 16. Mode permainan *credit game* dan *time survival* dijalankan

Keterangan dari nomor pada gambar adalah (1) mode permainan, (2) jumlah langkah saat ini, (3)Pilihan warna yang tersedia, (4) papan permainan, (5) papan terselesaikan (*time survival*), (6) waktu tersisa (*time survival*), (7) kredit langkah (*credit game*), dan (8) tingkat kesulitan (*credit game*).

Beberapa pengujian yang dilakukan untuk mengetahui apakah aplikasi permainan ini telah layak dipasarkan atau tidak antara lain uji kasus dan uji responden. Berikut adalah tabel hasil uji kasus terhadap aplikasi permainan. Deskripsi uji kasus diambil dari *Cheklist for designing better mobile games* [8].

Tabel 3. Hasil uji kasus aplikasi

Deskripsi	Kasus	Hasil yang diharapkan	Hasil yang didapat	Kesimpulan
Penyimpanan keadaan	Aplikasi tidak berhenti dan keluar saat home button ditekan	Saat home button ditekan aplikasi di minimize. Saat dibuka kembali aplikasi dapat dimainkan kembali.	Saat home button ditekan aplikasi di minimize dan saat dibuka kembali game sedang dalam keadaan pause	Diterima
Notifikasi	Suara permainan tidak menggangu suara notifikasi masuk.	Suara notifikasi dapat didengar walau permainan sedang dijalankan dengan suara.	Suara permainan lebih keras daripada suara notifikasi tetapi suara notifikasi masih dapat didengar	Diterima
Panggilan masuk	Panggilan masuk dapat diangkat saat sedang bermain	Permainan di-*minimize* dan panggilan dapat diangkat	Permainan di-*minimize* dan panggilan dapat diangkat. Setelah selesai permainan dapat dibuka dalam keadaan *pause*.	Diterima
Audio	Dapat dimainkan dengan atau tanpa suara	Dapat dimainkan dengan atau tanpa suara	*Terdapat interface* untuk mematikan suara pada menu utama dan *pause*.	Diterima
Audio	Aplikasi pemutar musik dapat dinyalakan saat permainan dibuka	Pemain dapat mendengarkan musik dari aplikasi lain saat sedang bermain	Saat suara permainan dimatikan pemain dapat bermain sambil mendengarkan musik dari aplikasi pemutar musik. Jika suara dihidupkan suara permainan dan pemutar musik bertubrukan.	Diterima
Audio	Suara mati saat permainan di-*minimize*.	Saat *home button* ditekan suara dan musik permainan berhenti dan saat dibuka kembali suara kembali menyala	Hasil yang didapat sama dengan hasil yang diharapkan saat pengaturan suara dinyalakan.	Diterima

Versi *alpha* dari aplikasi diluncurkan tanggal 25 Juni 2012 di forum *online* Android Medan dan forum Himpunan Mahasiswa Teknologi Informasi untuk menerima kritik dan saran. Sejak itu telah diterima banyak kritik dan masukan untuk pengembangan aplikasi ini. Perbaikan dan perubahan terhadap aplikasi yang kami buat merupakan kritik dan saran *beta tester* yang dapat ditampung dan diterima. Kami juga melakukan uji coba untuk mengukur tingkat kesulitan permainan. Hasil pengujian terhadap 10 responden dapat dilihat pada tabel 4 dan tabel 5.

Tabel 4. Hasil uji mode *credit game*

Responden	Square						Diamond					
	Easy		Medium		Hard		Easy		Medium		Hard	
	M	C	M	C	M	C	M	C	M	C	M	C
R 1	17	4	26	5	42	3	16	1	21	1	34	2
R 2	x	x	x	x	45	3	x	x	22	2	x	x
R 3	22	2	x	x	52	3	15	0	24	0	30	1
R 4	19	3	32	4	42	2	10	4	22	2	23	1
R 5	x	x	32	4	46	3	14	3	18	2	x	x
R 6	23	1	x	x	x	X	15	2	21	1	28	1
R 7	20	2	37	1	46	0	13	1	22	1	23	5
R 8	20	4	37	2	54	2	x	x	20	3	25	2
R 9	21	3	36	3	42	2	12	3	26	2	x	x
R 10	x	x	x	x	47	1	x	x	22	1	x	x

Tabel 5. Hasil uji mode *time survival*

Responden	Square		Diamond	
	M	F	M	F
R 1	29	3	30	5
R 2	43	4	67	8
R 3	22	3	35	5
R 4	41	4	55	7
R 5	30	3	41	6
R 6	25	3	42	6
R 7	17	2	40	6
R 8	24	3	30	5
R 9	23	3	67	7
R 10	48	5	55	7

Pada tabel, M adalah jumlah langkah yang dimainkan pemain sedangkan C adalah kredit langkah yang dimiliki *credit game*. Huruf F pada tabel adalah jumlah papan terselesaikan pada *time survival*. Dari hasil pengamatan pada permainan *credit* maka dapat dihitung tingkat keberhasilan mode permainan ini adalah :

Rumus : $keberhasilan = \frac{jumlah\ menang}{total\ main}\ x\ 100\%$ dan

$$rata - rata\ langkah = \frac{\sum_{i=0}^{n} langkah}{n}$$

1. *Square-Easy* : keberhasilan 70% dengan rata-rata langkah 20,29.
2. *Square-Medium* : keberhasilan 60% dengan rata-rata langkah 33,33.
3. *Square-Hard* : keberhasilan 90% dengan rata-rata langkah 46,22.
4. *Diamond-Easy* : keberhasilan 70% dengan rata-rata langkah 13,57.
5. *Diamond-Medium* : keberhasilan 100% dengan rata-rata langkah 21,8
6. *Diamond-Hard* : keberhasilan 60% dengan rata-rata langkah 27,16.

Sedangkan untuk mode permainan *time survival* adalah :

Rumus : $keberhasilan = \frac{jumlah\ menang}{total\ main}\ x\ 100\%$ dan

$$rata - rata\ papan\ selesai = \frac{\sum_{i=0}^{n} papan\ selesai}{n}$$

1. *Square* : keberhasilan 100% dengan rata-rata papan selesai 3,3.
2. *Diamond* : keberhasilan 100% dengan rata-rata papan selesai 6,2.

VI. KESIMPULAN

Telah dibangun aplikasi permainan *flood filling* bernama Floodit dengan menerapkan algoritma *flood fill* sebagai fungsi pewarnaan. Aplikasi yang dibangun telah melewati tinjauan *user requirement* dari perancangan aplikasi permainan *mobile* tanpa ditemukan kecacatan. Aplikasi permainan telah berhasil berjalan dibeberapa perangkat Android dengan resolusi dan ROM yang berbeda-beda.

Pengujian yang dilakukan terhadap beberapa responden untuk mengukur tingkat kesulitan pada mode *credit game* memiliki hasil persentasi kemenangan 70% *square-easy*, 60% *square medium*, 90% *square-hard*, 70% *diamond-easy*, 100% *diamond-medium*, dan 60% *diamond-hard*. Sedangkan untuk mode permainan *time survival* persentasi kemenangan adalah 100%.

Beberapa saran yang dapat dilakukan untuk penelitian lebih lanjut adalah pertimbangan mengenai tingkat kesulitan dari permainan ini. Pengembangan jenis papan dan mode permainan baru dapat juga dijadikan alternatif.

DAFTAR PUSTAKA

[1] Zechner, M. 2011. *Beginning Android Games*. New York: Apress.

[2] Munz, P., Hudea, I., Imad, J., dan Smith, R.J., 2009, When zombies attack!: Mathematical modeling of an outbreak of zombie infection. *Infectious Disease Modelling Research and Progress*: hal. 133-150. Nova Science Publishers.

[3] Kim, M. 2011. *Solving flood filling games with genetic algorithms*. Reno, Nevada : University of Nevada.

[4] Arthur, D., Clifford, R., Jalsenius, M., Montanaro, A., dan Sach, B. 2010. The complexity of flood filling games. *Fun with Algorithms, 5th International Confrence*: hal. 307-318.

[5] Meeks, K. dan Scott, A. 2011. *The complexity of flood filling games on graphs*. Oxford, Oxfordshire: University of Oxford.

[6] Meeks, K. dan Scott, A. 2012. *Spanning trees and the complexity of flood filling games*.

[7] Lagoutte, A., Noual, M., dan Thierry, E. 2011. *Flooding games on graphs*. Lyon : University de Lyon.

[8] Weedon, B. 2009. *Cheklist for Designing Better Mobile Games*. UK: PlayableGames.